

IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

APPLICATION FOR LETTERS PATENT

**Cryptography Correctness Detection Methods and
Apparatuses**

Inventors:

Monica Ene-Pietrosanu

Rajesh Ramadoss

Sermet Iskin

EV 395542183

ATTORNEY'S DOCKET NO. MS1-1762US

Cryptography Correctness Detection Methods and Apparatuses

TECHNICAL FIELD

The present invention relates generally to computers and like devices, and more particularly to methods and apparatuses that for detecting if cryptography information/services meet certain acceptable conditions from the security point of view for use by computing processes.

BACKGROUND

Cryptography services are typically provided in computing systems to support various security needs. These cryptography services employ different cryptography techniques and algorithms as needed to perform certain actions.

Cryptography techniques may be categorized as either symmetric cryptography or asymmetric cryptography. With symmetric cryptography, the same secret key is used for both encryption and decryption. This means that the symmetric key needs to be shared between the encrypting party and the decrypting party. Any party having a copy of the symmetric key may therefore decrypt and read a message. Hence, there is a need to protect and maintain control over the symmetric key. Security is provided through the protection of the key being used by the sender and the receiver. As long as only the sender and receiver know the secret symmetric key value, the message is protected (assuming a robust encryption algorithm and a cryptographically safe key size/seed are used).

Asymmetric cryptography (public key cryptography) is typically based on a "key pair". Here, one key in the pair is referred to as the "public" key. As the

1 name implies, a public key may be shared with others and even published in a
2 public directory, for example. The other key is referred to as the "private" key.
3 Also consistent with its name, the private key is meant to be kept secret and secure
4 by the party. Although the two keys are mathematically related, the private key
5 cannot be determined from the public key, or at least doing so would likely be
6 computationally infeasible.

7 Encryption and signing are two typical operations associated with public
8 key cryptography. Data that is encrypted using a public key can only be decrypted
9 using the associated private key and vice versa. Signing allows one to verify the
10 source of a piece of data. Signing does not, however, protect the data from being
11 viewed by anyone who has access to the sender's public key. In asymmetric
12 cryptography, security is provided through the protection of the private keys.

13 Asymmetric cryptography is also often employed to provide authentication,
14 non-repudiation and data integrity security mechanisms. Authentication provides
15 assurance that a message was actually sent by the party indicated. Non-
16 repudiation provides assurance that a sender cannot later deny having sent certain
17 data. Data Integrity provides assurance that a message was not modified prior to
18 reaching its destination.

19 These security mechanisms are typically provided by using a hash function
20 in conjunction with public key cryptography. A hash function is basically an
21 encoding scheme that is quick to compute and results in a relatively short numeric
22 representation of the message that was hashed. Hash functions can be used to
23 provide data integrity. First, a hash function is a one-way function, which means
24 that one cannot retrieve the message from the resulting hash value. Second, the
25

1 slightest change to the original message will result in a clearly detectable change
2 of the hash value.

3 Some processes use a hash function in conjunction with public key
4 cryptography to provide a security service often referred to as “signing” that
5 ensures authentication and non-repudiation. For example, in certain systems,
6 when a user signs a message, a hash of the message is calculated and then
7 encrypted using the sender's private key. The resulting encrypted hash is referred
8 to as the “digital signature”. The original plaintext message, the digital signature,
9 and the sender's certificate which contains the sender's public signing key are then
10 sent to the recipient. Once received, the digital signature is decrypted using the
11 sender's public key that was sent along with the message in the form of a
12 certificate. The receiving client also generates a hash value for the plaintext
13 message using the same hash function as did the sender. After the signature of the
14 sender is decrypted with the sender's public key and the hash value recovered, the
15 recovered hash value can then be compared with the generated hash value to
16 detect differences. If the two hash values match, then the message must have
17 originated from the sender who possesses the private key. Hence, this provides
18 authentication and non-repudiation. Furthermore, since this technique reliably
19 detects if the message was changed/tampered during transit, data integrity is
20 provided.

21 Cryptography services such as these and others are often handled
22 “automatically” by the processes running on computing devices. This means,
23 however, that such processes and/or users are sometimes not aware of the type of
24 algorithm/key being used, nor if such algorithms/keys may be less secure than
25 others that are available for use.

1 Consequently, for such reasons and others, there is a need for methods and
2 apparatuses that can inform certain processes and/or even the user about the
3 relative strength/weakness of cryptography services being used.

4 5 SUMMARY

6 Methods and apparatuses are provided that can inform certain processes
7 and/or even the user about the relative strength/weakness of cryptography services
8 being used.

9 The above stated needs and/or others are met, for example, by a method
10 that includes establishing at least one cryptography service parameter threshold,
11 selectively detecting a request for at least one cryptography service, and
12 selectively performing at least one correctness detection action based on the
13 requested cryptography service and the cryptography service parameter threshold.

14 The cryptography service parameter threshold may identify
15 acceptable/unacceptable cryptography algorithms, acceptable/unacceptable
16 cryptography key size parameters, acceptable/unacceptable cryptography seed size
17 parameters, and other like parameters with which requested cryptography service
18 information can be compared.

19 Algorithms, for example, may be categorized as being certified, old/out-of-
20 date, weak, strong, etc. Key/seed lengths may also be compared to threshold
21 lengths that are considered either weak or strong.

22 In certain implementations the method may also include performing
23 actions, such as, for example, interrupting the application process, stopping the
24 application process, starting at least one process to do further actions, displaying
25 alert information, logging alert information, suggesting at least one alternative

1 cryptography service, outputting alert messages, causing alteration of a graphical
2 user interface, forcing use of at least one other cryptography service, etc., if the
3 requested cryptography service is deemed to be "too weak".
4

5 **BRIEF DESCRIPTION OF THE DRAWINGS**

6 A more complete understanding of the various methods and apparatuses of
7 the present invention may be had by reference to the following detailed description
8 when taken in conjunction with the accompanying drawings wherein:

9 Fig. 1 is a block diagram that depicts a computer system configurable to
10 provide cryptography services and cryptography correctness detection logic.

11 Fig. 2 is a block diagram depicting an example of certain computing
12 processes including exemplary cryptography correctness detection logic and
13 cryptography algorithm logic, for use in a computing device, for example, as
14 depicted in Fig. 1.

15 Fig. 3 is a flow diagram depicting certain exemplary acts associated with an
16 exemplary method for use in cryptography correctness detection logic, for
17 example, as in Fig. 2.

18 Fig. 4 is a flow diagram depicting certain exemplary acts associated with an
19 exemplary method for use in cryptography correctness detection logic when
20 asymmetric key encryption is requested.

21 Fig. 5 is a flow diagram depicting certain exemplary acts associated with an
22 exemplary method for use in cryptography correctness detection logic when
23 symmetric key encryption is requested.
24
25

1 Fig. 6 is a flow diagram depicting certain exemplary acts associated with an
2 exemplary method for use in cryptography correctness detection logic when
3 asymmetric key decryption is requested.

4 Fig. 7 is a flow diagram depicting certain exemplary acts associated with an
5 exemplary method for use in cryptography correctness detection logic when
6 symmetric key decryption is requested.

7 Fig. 8 is a flow diagram depicting certain exemplary acts associated with an
8 exemplary method for use in cryptography correctness detection logic when a new
9 key is derived.

10 Fig. 9 is a flow diagram depicting certain exemplary acts associated with an
11 exemplary method for use in cryptography correctness detection logic when a key
12 is imported or exported.

13 14 **DETAILED DESCRIPTION**

15 Turning to the drawings, wherein like reference numerals refer to like
16 elements, the invention is illustrated as being implemented in a suitable computing
17 environment. Although not required, the invention will be described in the general
18 context of computer-executable instructions, such as program modules, being
19 executed by a personal computer. Generally, program modules include routines,
20 programs, objects, components, data structures, etc. that perform particular tasks
21 or implement particular abstract data types. Moreover, those skilled in the art will
22 appreciate that the invention may be practiced with other computer system
23 configurations, including hand-held devices, multi-processor systems,
24 microprocessor based or programmable consumer electronics, network PCs,
25 minicomputers, mainframe computers, and the like. The invention may also be

1 practiced in distributed computing environments where tasks are performed by
2 remote processing devices that are linked through a communications network. In
3 a distributed computing environment, program modules may be located in both
4 local and remote memory storage devices.

5 Fig.1 illustrates an example of a suitable computing environment 120 with
6 which the subsequently described methods and apparatuses may be implemented.

7 Exemplary computing environment 120 is only one example of a suitable
8 computing environment and is not intended to suggest any limitation as to the
9 scope of use or functionality of the improved methods and apparatuses described
10 herein. Neither should computing environment 120 be interpreted as having any
11 dependency or requirement relating to any one or combination of components
12 illustrated in computing environment 120.

13 The improved methods and apparatuses herein are operational with
14 numerous other general purpose or special purpose computing system
15 environments or configurations. Examples of well known computing systems,
16 environments, and/or configurations that may be suitable include, but are not
17 limited to, personal computers, server computers, thin clients, thick clients, hand-
18 held or laptop devices, multiprocessor systems, microprocessor-based systems, set
19 top boxes, programmable consumer electronics, network PCs, minicomputers,
20 mainframe computers, distributed computing environments that include any of the
21 above systems or devices, and the like.

22 As shown in Fig. 1, computing environment 120 includes a general-purpose
23 computing device in the form of a computer 130. The components of computer
24 130 may include one or more processors or processing units 132, a system
25

1 memory 134, and a bus 136 that couples various system components including
2 system memory 134 to processor 132.

3 Bus 136 represents one or more of any of several types of bus structures,
4 including a memory bus or memory controller, a peripheral bus, an accelerated
5 graphics port, and a processor or local bus using any of a variety of bus
6 architectures. By way of example, and not limitation, such architectures include
7 Industry Standard Architecture (ISA) bus, Micro Channel Architecture (MCA)
8 bus, Enhanced ISA (EISA) bus, Video Electronics Standards Association (VESA)
9 local bus, and Peripheral Component Interconnects (PCI) bus also known as
10 Mezzanine bus.

11 Computer 130 typically includes a variety of computer readable media.
12 Such media may be any available media that is accessible by computer 130, and it
13 includes both volatile and non-volatile media, removable and non-removable
14 media.

15 In Fig. 1, system memory 134 includes computer readable media in the
16 form of volatile memory, such as random access memory (RAM) 140, and/or non-
17 volatile memory, such as read only memory (ROM) 138. A basic input/output
18 system (BIOS) 142, containing the basic routines that help to transfer information
19 between elements within computer 130, such as during start-up, is stored in ROM
20 138. RAM 140 typically contains data and/or program modules that are
21 immediately accessible to and/or presently being operated on by processor 132.

22 Computer 130 may further include other removable/non-removable,
23 volatile/non-volatile computer storage media. For example, Fig. 1 illustrates a
24 hard disk drive 144 for reading from and writing to a non-removable, non-volatile
25 magnetic media (not shown and typically called a "hard drive"), a magnetic disk

1 drive 146 for reading from and writing to a removable, non-volatile magnetic disk
2 148 (e.g., a "floppy disk"), and an optical disk drive 150 for reading from or
3 writing to a removable, non-volatile optical disk 152 such as a CD-ROM, CD-R,
4 CD-RW, DVD-ROM, DVD-RAM or other optical media. Hard disk drive 144,
5 magnetic disk drive 146 and optical disk drive 150 are each connected to bus 136
6 by one or more interfaces 154.

7 The drives and associated computer-readable media provide nonvolatile
8 storage of computer readable instructions, data structures, program modules, and
9 other data for computer 130. Although the exemplary environment described
10 herein employs a hard disk, a removable magnetic disk 148 and a removable
11 optical disk 152, it should be appreciated by those skilled in the art that other types
12 of computer readable media which can store data that is accessible by a computer,
13 such as magnetic cassettes, flash memory cards, digital video disks, random access
14 memories (RAMs), read only memories (ROM), and the like, may also be used in
15 the exemplary operating environment.

16 A number of program modules may be stored on the hard disk, magnetic
17 disk 148, optical disk 152, ROM 138, or RAM 140, including, e.g., an operating
18 system 158, one or more application programs 160, other program modules 162,
19 and program data 164.

20 The improved methods and apparatuses described herein may be
21 implemented within operating system 158, one or more application programs 160,
22 other program modules 162, and/or program data 164.

23 A user may provide commands and information into computer 130 through
24 input devices such as keyboard 166 and pointing device 168 (such as a "mouse").
25 Other input devices (not shown) may include a microphone, joystick, game pad,

1 satellite dish, serial port, scanner, camera, etc. These and other input devices are
2 connected to the processing unit 132 through a user input interface 170 that is
3 coupled to bus 136, but may be connected by other interface and bus structures,
4 such as a parallel port, game port, or a universal serial bus (USB).

5 A monitor 172 or other type of display device is also connected to bus 136
6 via an interface, such as a video adapter 174. In addition to monitor 172, personal
7 computers typically include other peripheral output devices (not shown), such as
8 speakers and printers, which may be connected through output peripheral interface
9 175.

10 Computer 130 may operate in a networked environment using logical
11 connections to one or more remote computers, such as a remote computer 182.
12 Remote computer 182 may include many or all of the elements and features
13 described herein relative to computer 130.

14 Logical connections shown in Fig. 1 are a local area network (LAN) 177
15 and a general wide area network (WAN) 179. Such networking environments are
16 commonplace in offices, enterprise-wide computer networks, intranets, and the
17 Internet.

18 When used in a LAN networking environment, computer 130 is connected
19 to LAN 177 via network interface or adapter 186. When used in a WAN
20 networking environment, the computer typically includes a modem 178 or other
21 means for establishing communications over WAN 179. Modem 178, which may
22 be internal or external, may be connected to system bus 136 via the user input
23 interface 170 or other appropriate mechanism.

1 Depicted in Fig. 1, is a specific implementation of a WAN via the Internet.
2 Here, computer 130 employs modem 178 to establish communications with at
3 least one remote computer 182 via the Internet 180.

4 In a networked environment, program modules depicted relative to
5 computer 130, or portions thereof, may be stored in a remote memory storage
6 device. Thus, e.g., as depicted in Fig. 1, remote application programs 189 may
7 reside on a memory device of remote computer 182. It will be appreciated that the
8 network connections shown and described are exemplary and other means of
9 establishing a communications link between the computers may be used.

10 Attention is now drawn to Fig. 2, which is a block diagram illustratively
11 depicting certain computer-implementable processes 200 that are configured to
12 operate together in a manner such that cryptographic services can be requested by
13 certain processes and provided by other processes to the requesting process
14 accordingly. The cryptographic services may include, for example, encryption
15 services, decryption services, key generation services, key exchange services,
16 digital signature services, etc. Typically, cryptographic services are requested or
17 otherwise initiated by processes, such as applications, and the actual cryptography
18 services performed by one or more other cryptography algorithm processes. The
19 applications usually interact with the operating system through the application
20 programming interfaces (API) to initiate the cryptography services.

21 Thus, for example, as depicted in Fig. 2 an operating system 202 contains
22 interfaces (APIs) between an application 208 and cryptography algorithm logic
23 206. Included in operating system 202 is cryptography correctness detection logic
24 204. While illustrated in this example as being part of the operating system, in
25 other implementations cryptography correctness detection logic 204 may be a

1 process that is separate from operating system 202 but operatively coupled to
2 operating system 202 to perform functions/acts accordingly.

3 Attention is now drawn to Fig. 3, which is a flow diagram showing a
4 method 300 that cryptography correctness detection logic 204 in the example in
5 Fig. 2 can be configured to perform.

6 In act 302, which is optional, cryptography correctness parameter(s) and/or
7 other like information are updated and maintained. This may include an initial
8 establishment of the cryptography correctness parameter(s) and/or other like
9 information, or the periodic or otherwise selective updating of cryptography
10 correctness parameter(s) and/or other like information. As a result of act 302,
11 cryptography correctness parameter(s) and/or other like information is configured
12 and stored, for example, in a computer's memory. These cryptography correctness
13 parameters may be configured according to the security requirements for the
14 application and may change in time according to the computational power available
15 and advances made in cryptanalysis.

16 In act 304, the current relative "strength" for each available cryptography
17 algorithm logic 206 is established. Here, for example, the cryptography
18 correctness parameter(s) and/or other like information as provided in act 302 may
19 include one or more cryptography service parameter thresholds. Such parameter
20 thresholds can identify acceptable ("strong" enough) and/or unacceptable (too
21 "weak") cryptography algorithms, or acceptable/unacceptable cryptography key
22 size parameters. Cryptography algorithms can be specified using algorithm
23 identifiers, version numbers, etc., and cryptography key size parameters can be
24 identified by specifying acceptable/unacceptable bit lengths, for example.

1 In certain implementations, as part of acts 302/304, the cryptography
2 service parameter threshold(s) are further associated with correctness categories.
3 These correctness categories can, for example be employed to define the different
4 algorithm identifiers, key lengths, etc., as being “old”/outdated algorithms,
5 new/strong algorithms, weak keys, and strong keys.

6 The cryptography service parameter threshold may also identify
7 acceptable/unacceptable seed type/size parameters associated with cryptography
8 services, such as key generation.

9 In act 306, cryptography correctness detection logic 204 is configured to
10 monitor on-going applicable processes to detect or otherwise be made aware of a
11 request for or use of cryptography services from cryptography algorithm logic
12 206. For example, in certain implementations application 208 alerts operating
13 system 202 as to a need for cryptography services. Cryptography correctness
14 detection logic 204 is made aware of this request in act 306.

15 In act 306, various processes may be monitored; for example, application
16 processes, operating system services, managed code application processes, or
17 other processes calling into the cryptographic application programming interfaces
18 (API) processes, and/or the like can be monitored.

19 In act 308, cryptography correctness detection logic 204 determines if the
20 requested cryptography service/algorithm meets the conditions established in acts
21 302/304 via the cryptography correctness parameters/information. For example,
22 in act 308 it can be determined if the identified cryptography algorithm is
23 considered to be “strong” enough or too “weak” for a given process, time, user,
24 data, etc. This may include, for example, determining a category for the
25 algorithm/key. This may also include determining the type/length of a key to be

1 used and comparing the type/length to applicable cryptography correctness
2 parameters/information.

3 If, in act 308, it is determined that the requested cryptography
4 service/algorithm satisfies the applicable cryptography correctness
5 parameters/information, then the cryptography service/algorithm continues to
6 execute accordingly. Information may be logged by cryptography correctness
7 detection logic regarding the monitoring activities in act 306 and/or the
8 determination made in act 308.

9 To the contrary, if, in act 308, it is determined that the requested
10 cryptography service/algorithm fails to satisfy the applicable cryptography
11 correctness parameters/information, then the cryptography service/algorithm
12 continues with act 310.

13 In act 310, one or more actions may be initiated or otherwise performed by
14 cryptography correctness detection logic 204. By way of example, actions may
15 include interrupting the application process, stopping the application process,
16 starting at least one process to perform further correction/notification actions,
17 displaying alert information, logging alert information, suggesting at least one
18 alternative cryptography service, outputting alert messages, causing alteration of a
19 graphical user interface, and/or forcing use of at least one other cryptography
20 algorithm/service instead of the requested algorithm/service.

21 In this manner, cryptography correctness detection logic 204 can be
22 configured to support or enforce specific security/policy requirements depending
23 on the device/user/situation. Thus, in certain implementations, when a “weak”
24 algorithm/key is detected by cryptography correctness detection logic 204, then
25 the algorithm/key may be flagged accordingly to alert the program, user,

1 administrator, etc., about the use of a weak algorithm/key/seed, while also
2 allowing the requested cryptography service to continue. In other stricter
3 implementations, a requested cryptography service that is deemed to be too weak
4 may not be continued or otherwise refused to occur. In still other examples, an
5 implementation may actively suggest one or more different, i.e., “stronger”,
6 algorithms/keys/seeds. Here, a user can be presented with and selectively
7 authorize such substitution. In still other implementations, such a substitution may
8 be made automatically or at least initiated automatically by cryptography
9 correctness detection logic 204. Information may also be logged by cryptography
10 correctness detection logic 204 regarding the actions initiated in act 310.

11 Figs 4-9 provide additional description for use in logic 204 and/or method
12 300 in accordance with certain further illustrative examples.

13 Fig. 4 is a flow diagram depicting certain exemplary acts associated with an
14 exemplary method 400 for use in cryptography correctness detection logic 204
15 when asymmetric/public key encryption is requested. In act 402, a request for
16 encryption services with a public key is made by application 208, for example. In
17 act 404, it is determined if the public key is sufficiently secure enough for the
18 present operation. For example, in act 404 the size of the public key may be
19 compared to a minimum acceptable public key size (e.g., greater than or equal to
20 N bits, with N currently equal to 1024). In act 406, the public key is sufficiently
21 secure and therefore control is returned to the application or other applicable
22 process. In act 408, the public key is deemed to be not sufficiently secure and
23 therefore action is initiated in the form of a flag weak key action. In act 410,
24 processes continue to execute accordingly.
25

Fig. 5 is a flow diagram depicting certain exemplary acts associated with an exemplary method 500 for use in cryptography correctness detection logic 204 when symmetric key encryption is requested. In act 502, a request for encryption services with a symmetric key is made by application 208, for example. In act 504, it is determined if the cryptography algorithm is an RC2 algorithm. In act 506, the algorithm is not the RC2 algorithm, and it is determined if the symmetric key is sufficiently secure, e.g., based on its size (length in bits) (e.g., less than K bits, with K currently equal to 128). If the symmetric key is deemed sufficiently secure, then control is returned to the application or other applicable process in act 516. If the symmetric key is deemed to not be sufficiently secure, then in act 508 action is initiated in the form of a flag weak key action and the method continues with act 516.

Retuning to act 504, if the algorithm is determined to be RC2, then in method 500 continues with act 510, in which action is initiated in the form of a flag old algorithm action. This action may include recommending a substitute algorithm. In act 512, it is determined if the effective key size is sufficiently secure, e.g., based on its size (length in bits) (e.g., less than M bits, with M currently equal to 128). If the symmetric key is deemed sufficiently secure, then control is returned to the application or other applicable process in act 516. If the symmetric key is deemed to not be sufficiently secure, then in act 514 action is initiated in the form of a flag weak key action and the method continues with act 516.

Fig. 6 is a flow diagram depicting certain exemplary acts associated with an exemplary method 600 for use in cryptography correctness detection logic 204 when asymmetric key decryption is requested. In act 602, a request for decryption

1 services with a public key is made by application 208, for example. In act 604, it
2 is determined if the public key is sufficiently secure enough for the present
3 operation. For-example, in act 604 the size of the public key may be compared to
4 a minimum acceptable public key size (e.g., greater than or equal to N bits, with N
5 currently equal to 1024). In act 606, the public key is sufficiently secure and
6 therefore control is returned to the application or other applicable process. In act
7 608, the public key is deemed to be not sufficiently secure and therefore action is
8 initiated in the form of a flag weak key action. In act 610, processes continue to
9 execute accordingly.

10 Fig. 7 is a flow diagram depicting certain exemplary acts associated with an
11 exemplary method 700 for use in cryptography correctness detection logic 204
12 when symmetric key decryption is requested. In act 702, a request for decryption
13 services with a symmetric key is made by application 208, for example. In act
14 704, it is determined if the cryptography algorithm is an RC2 algorithm. If the
15 algorithm is not RC2, then method 700 continues with act 712 and processes
16 continue to execute accordingly. If the algorithm is RC2, then method 700
17 continues with act 706.

18 In act 706 action is initiated in the form of a flag old algorithm action. This
19 action may include recommending a substitute algorithm. In act 708, it is
20 determined if the effective key size is sufficiently secure, e.g., based on its size
21 (length in bits) (e.g., less than M bits, with M currently equal to 128). If the
22 symmetric key is deemed sufficiently secure, then control is returned to the
23 application or other applicable process in act 712. If the symmetric key is deemed
24 to not be sufficiently secure, then in act 710 action is initiated in the form of a flag
25

1 weak ciphertext may have been compromised action and the method continues
2 with act 712.

3 Fig. 8 is a flow diagram depicting certain exemplary acts associated with an
4 exemplary method 800 for use in cryptography correctness detection logic 204
5 when deriving a new key (e.g., based on a seed). In act 802, a request for
6 encryption services providing a new key is made by application 208, for example.
7 In act 804, it is determined if the initial key (or seed) from which the new key is to
8 be derived from is sufficiently secure enough for the present operation. For
9 example, in act 804 the size of the initial key or seed may be compared to a
10 minimum acceptable initial key or seed size. In act 806, the initial key or seed is
11 sufficiently secure and therefore control is returned to the application or other
12 applicable process. In act 808, the initial key or seed is deemed to be not
13 sufficiently secure and therefore action is initiated in the form of a flag weak key
14 action. In act 810, processes continue to execute accordingly.

15 Fig. 9 is a flow diagram depicting certain exemplary acts associated with an
16 exemplary method 900 for use in cryptography correctness detection logic 204
17 when a key is imported or exported. In act 902, a request for encryption services
18 includes importing a key. In act 904, a request for encryption services includes
19 exporting a key.

20 In act 906, it is determined if a public key that is used for key
21 encryption/decryption of the imported/exported key is sufficiently secure enough
22 for the present operation. For example, in act 906 the size of the public key may
23 be compared to a minimum acceptable public key size. If, in act 908, the public
24 key is deemed to be sufficiently secure, then method 900 continues with act 910.
25 If, in act 908, the public key is deemed to not be sufficiently secure, then method

1 900 continues with act 908 and action is initiated in the form of a flag weak key
2 action because the imported/exported key may have been (or may become)
3 exposed or otherwise more easily compromised.

4 In act 910, it is determined if an imported key is sufficiently secure enough
5 for the present operation. For example, in act 910 the size of the imported key
6 may be compared to a minimum acceptable importable key size. If, in act 910, the
7 imported key is deemed to be sufficiently secure, then method 900 continues with
8 act 914. If, in act 908, the imported key is deemed to not be sufficiently secure,
9 then method 900 continues with act 912 and action is initiated in the form of a flag
10 weak key action because the imported key may have been exposed or otherwise
11 more compromised.

12 In act 914, it is determined if an imported/exported key is an RC2 key. If,
13 in act 914 the imported/exported key is not an RC2 key, then method 900
14 continues with act 918. If the imported/exported key is an RC2 key, then method
15 900 continues with act 916 and action is initiated in the form of a flag key action.
16 This action may include recommending a substitute key/algorithm. In act 918,
17 processes continue to execute accordingly.

18 In this section some current cryptography algorithms are identified by
19 common their request calls and/or name. Some of these algorithms, for example,
20 are already deemed to be less secure (weak) when compared to others that are
21 currently considered "strong". Those skilled in the art will clearly recognize that
22 this exemplary list may be increased or decreased in size and the suggested
23 relative strengths of the algorithms will likely need to change over time as new
24 developments in the field of cryptography are developed.
25

1	Algorithm internal identifier	Message displayed and recommendation
	CALG_MD2	MD2 hashing (weak algorithm)
2	CALG_MD4	MD4 Hashing (weak algorithm)
	CALG_MD5	MD5 Hashing (strong algorithm)
3	CALG_SHA1	SHA1 Hashing (strong algorithm)
	CALG_MAC	MAC Hashing
4	CALG_RSA_SIGN	RSA signing
	CALG_DSS_SIGN	DSS signing
5	CALG_NO_SIGN	CALG_NO_SIGN: No signature
	CALG_RSA_KEYX	RSA key exchange
6	CALG_DES	DES (weak encryption)
7	CALG_3DES_112	3DES-2 key encryption (strong algorithm)
	CALG_3DES	3DES-3 key (strong algorithm)
8	CALG_DESX	DESX encryption
9	CALG_RC2	RC2 (old encryption, use newer if possible)
	CALG_RC4	RC4 (strong algorithm)
10	CALG_RC5	RC5 (weak algorithm)
	CALG_AES_128	AES_128 (strong algorithm)
11	CALG_AES_192	AES_192 (strong algorithm)
	CALG_AES_256	AES_256 (strong algorithm)
12	CALG_AES	Generic AES encryption (strong algorithm)
13	CALG_SEAL	CALG_SEAL encryption
	CALG_DH_SF	CALG_DH_SF: Diffie-Hellman
14		(store and forward) key agreement
15	CALG_DH_EPHEM	CALG_DH_EPHEM: Diffie-Hellman
		(ephemeral) key agreement"},
16	CALG_AGREEDKEY_ANY	CALG_AGREEDKEY_ANY:
		(any other key agreement)
17	CALG_KEA_KEYX	CALG_KEA_KEYX: KEA key exchange"},
	CALG_HUGHES_MD5	CALG_HUGHES_MD5:
18		(Hughes MD5 hashing)
19	CALG_SKIPJACK	CALG_SKIPJACK: Skipjack encryption
	CALG_TEK	CALG_TEK: TEK encryption
20	CALG_CYLINK_MEK	CALG_CYLINK_MEK:
		(Cylink MEK encryption)
21	CALG_SSL3_SHAMD5	CALG_SSL3_SHAMD5: (for SSL3)
	CALG_SSL3_MASTER	CALG_SSL3_MASTER:
22		(Master key encryption for SSL3)
23	CALG_SCHANNEL_	
	MASTER_HASH	CALG_SCHANNEL_MASTER_HASH
24		(Master key hashing for Schannel)
25	CALG_SCHANNEL_	

1	MAC_KEY	CALG_SCHANNEL_MAC_KEY (MAC for Schannel)
2	CALG_SCHANNEL_ ENC_KEY	CALG_SCHANNEL_ENC_KEY
3	CALG_PCT1_MASTER	CALG_PCT1_MASTER (old algorithm, suggest upgrade to newer)
4	CALG_SSL2_MASTER	CALG_SSL2_MASTER (weak algorithm)
5	CALG_TLS1_MASTER	CALG_TLS1_MASTER (strong algorithm)
6	CALG_HMAC	CALG_HMAC: MAC with key
7	CALG_TLS1PRF	CALG_TLS1PRF
8	CALG_HASH_ REPLACE_OWF	CALG_HASH_REPLACE_OWF
9	CALG_SHA_256	CALG_SHA_256 hashing for AES (strong algorithm)
10	CALG_SHA_384	CALG_SHA_384 hashing for AES (strong algorithm)
11	CALG_SHA_512	CALG_SHA_512 hashing for AES (strong algorithm)

Although some preferred implementations of the various methods and apparatuses have been illustrated in the accompanying Drawings and described in the foregoing Detailed Description, it will be understood that the invention is not limited to the exemplary implementations disclosed, but is capable of numerous rearrangements, modifications and substitutions without departing from the spirit of the invention as set forth and defined by the following claims.